APPLICATION FOR UNITED STATES PATENT

in the name of

Changmin Liu
Yan Ke

of

Netscreen Technologies, Inc.

for

Guaranteed Bandwidth Sharing in a Traffic Shaping System

Mark D. Kirkland, Reg. No. 40,048
Fish & Richardson P.C.
2200 Sand Hill Road, Suite 100
Menlo Park, CA 94025
Tel.: (650) 322-5070
Fax: (650) 854-0875

ATTORNEY DOCKET:
09725-005001

DATE OF DEPOSIT: September 8, 2000
EXPRESS MAIL NO.: EL 572620444 US

GUARANTEED BANDWIDTH SHARING IN A TRAFFIC SHAPING SYSTEM

TECHNICAL FIELD

The present invention relates generally to electrical computing systems, and more particularly to tools for shaping network traffic in a communication system.

BACKGROUND

There are many emerging trends in the communications world, including the increase in network technology and the proliferation of data networks. To ensure security of communications and to avoid networking congestion problems, network designers have either incorporated security appliances, such as firewalls and virtual private networks, and traffic management devices in their systems or enhanced their routers with these functionalities.

A firewall is an Internet security appliance designed to screen traffic coming into and out of a network location. A virtual private network provides a secure connection through a public network such as the Internet, between two or more distant network appliances using a virtual private networking technology. Traffic management devices are used to monitor and/or control traffic passing through network devices to maintain a network's quality-of-service (QOS) level.

Traffic shaping is a technique for regulating traffic. Traffic shaping refers to a process of analyzing an allocated bandwidth utilized by various types of network traffic through a network appliance. Traffic shaping policies can be used to prioritize users in accordance with an IP address or other criteria to provide different levels of service to different users of a network appliance.

A leaky bucket is one example of a conventional technique used to control traffic flow. A splash is added to the bucket for each incoming byte received by the network appliance when the bucket is not full. The splash results in an increment in a counter associated with the bucket. The bucket leaks away at a constant rate (bytes are allowed to pass from the bucket through the network appliance). When the bucket is full (the counter is at a maximum value), bytes cannot pass through the network appliance (i.e., no additional bytes can be added to the bucket). As the bucket begins to empty in accordance with the leak

rate, bytes once again can begin to be added to the bucket so that they can be moved through the network appliance. Leaky buckets are characterized by three parameters: the sustained information rate (SIR), the peak information rate (PIR) and the burst length (BL). The SIR is bound by the long-term data rate and defines the rate that data leaks from the bucket. BL defines the depth of the bucket and is equal to the maximum amount of data allowed to be passed when the bucket is empty. PIR defines the maximum rate at which the source can send data.

A token bucket is another example of a conventional technique used to control traffic flow. With token buckets, tokens are added to the bucket at a constant rate of the SIR. The token is a convention that is used to represent a defined unit of data (e.g., a byte or a packet). The token bucket has a depth of BL. When the token bucket is full no more tokens can be added to the bucket. When data comes in to the network appliance, if there are enough tokens in the token bucket, the data will be passed. Otherwise, the data will be held until the token bucket accumulates enough tokens.

A credit/debit token bucket is another example of a conventional technique used to control traffic flow. In the example discussed above for the token bucket, no data can be passed when there are insufficient tokens in the token bucket. This may cause some unacceptable delays. A credit/debit token bucket allows for oversubscription of tokens. Unlike normal token buckets, the number of tokens in a credit/debit token bucket can be negative. Based on the number of tokens in the bucket, a credit/debit token bucket can be in either one of the three states. If the number of tokens is positive the bucket has credit. If the number of tokens is negative the bucket owes a debit. Otherwise the bucket is in an even state. The operation of a conventional credit/debit token bucket is as follows. Tokens are added to a credit/debit token bucket in the same way as added to a normal token bucket. The difference between a normal token bucket and credit/debit token bucket is how incoming data is handled. In a credit/debit token bucket, as long as the data comes in and the credit/debit token bucket is in the credit or even state, the data is granted permission to pass and the number of tokens is decremented accordingly. After the decrementing operation (to remove the corresponding number of tokens from the bucket based on the amount of data that was passed), the number of tokens in the bucket can be negative, even, or still positive. If data comes in while the credit/debit token bucket is in a debit state, the data is held until the

credit/debit bucket returns to the credit or even state (i.e., pays off all of its debt). Thereafter, the network appliance can pass the held data.

As described above a bucket has a definite depth. For example, a token bucket can accept a definite number of tokens before it becomes full. Traditionally, when the bucket became full, the excess (e.g., the excessive tokens) was discarded (i.e., to satisfy a guaranteed bandwidth requirement). If an unlimited or undefined depth bucket were used then the underlying policy may become oversubscribed. In order to ensure that oversubscription does not occur, traditionally these excessive tokens were discarded. What would be desirable is a system that would allow a bucket to share its excess while still providing a guaranteed bandwidth to an associated policy.

SUMMARY

In one aspect the invention provides a method for allocating bandwidth in a network appliance where the network appliance includes a plurality of guaranteed bandwidth buckets used to evaluate when to pass traffic through the network appliance. The method includes providing a shared bandwidth bucket associated with a plurality of the guaranteed bandwidth buckets, allocating bandwidth to the shared bandwidth bucket based on the underutilization of bandwidth in the plurality of guaranteed bandwidth buckets and sharing excess bandwidth developed from the underutilization of the guaranteed bandwidth allocated to the individual guaranteed bandwidth buckets. The step of sharing includes borrowing bandwidth from the shared bandwidth bucket by a respective guaranteed bandwidth bucket to allow traffic to pass immediately through the network appliance.

Aspects of the invention can include one or more of the following features. The shared bandwidth bucket can be a token bucket. The guaranteed bandwidth buckets can be token buckets or credit/debit buckets. Each guaranteed bandwidth bucket can be associated with a traffic shaping policy. A plurality of guaranteed bandwidth buckets can be associated with a single traffic shaping policy. The traffic shaping policy can screen based on IP address. The traffic shaping policy can screen based on the source IP address, the destination IP address, the protocol type, UPD/TCP port number, type of service requested and/or the traffic content.

In another aspect the invention provides a method for allocating bandwidth in a network appliance. The method includes defining a guaranteed bandwidth allocation for a first policy for passing traffic through the network appliance including using a first bucket to allocate the guaranteed bandwidth. A guaranteed bandwidth allocation for a second policy for passing traffic through the network appliance is also defined including using a second bucket to allocate the guaranteed bandwidth. Excess bandwidth developed from the underutilization of the guaranteed bandwidth allocated to the first and second buckets is shared. Sharing includes providing a shared bandwidth bucket associated with first and second buckets and borrowing bandwidth from the shared bandwidth bucket by one of the first and second buckets when the respective bucket has insufficient bandwidth to allow traffic to pass immediately through the network appliance.

In another aspect, the invention provides an apparatus for allocating bandwidth in a network appliance where the network appliance includes a plurality of guaranteed bandwidth buckets used to evaluate when to pass traffic through the network appliance. The apparatus includes a shared bandwidth bucket associated with a plurality of the guaranteed bandwidth buckets, means for allocating bandwidth to the shared bandwidth bucket based on the underutilization of bandwidth in the plurality of guaranteed bandwidth buckets and a scheduler. The scheduler is operable to evaluate a packet to determine if a traffic shaping policy should be applied to a given packet, evaluate a guaranteed bandwidth bucket associated with an identified traffic shaping policy, determine when the guaranteed bandwidth bucket associated with an identified traffic shaping policy has insufficient capacity to support a transfer of the packet through the network and borrow bandwidth from the shared bandwidth bucket by a respective guaranteed bandwidth bucket to allow traffic to pass immediately through the network appliance.

Aspects of the invention can include one or more of the following advantages. Cascaded buckets are provided that support a guaranteed bandwidth for a policy while also allowing for sharing of that guaranteed bandwidth among other policies in the event that the bandwidth is under utilized. The shared bandwidth supports the elimination of static bandwidth allocation while guaranteed bandwidth is not compromised. By allowing the network appliance to share the unused guaranteed bandwidth, the bandwidth utilization of the network appliance can be dramatically improved.

The details of one or more embodiments of the invention are set forth in the accompanying drawings and the description below. Other features, objects, and advantages of the invention will be apparent from the description and drawings, and from the claims.

DESCRIPTION OF DRAWINGS

FIG. 1 is a shows a block diagram for a network appliance that includes cascaded buckets.

FIG. 2 is a flow diagram of a process for passing data through the network appliance of FIG. 1.

FIG. 3 is a flow diagram of a method for incrementing a guaranteed bandwidth bucket.

Like reference symbols in the various drawings indicate like elements.

DETAILED DESCRIPTION

Referring now to FIG. 1, a network appliance 100 that includes traffic shaping policy tools is shown. Network appliance 100 includes a plurality of input ports 102 each of which includes an input queue 104, an output queue 106, a scheduler 108 and a forwarding engine 120. Associated with scheduler 108 are a token system 110, increment engine 112 and decrement engine 114.

As data is received at an input port 102, the data is transported into the input queue 104. Input queue 104 acts as a first in/first out buffer for holding packets that are received from the input port 102. Scheduler 108 periodically checks the status of token system 110 to determine whether or not a data packet stored in the input queue 104 can be passed to the output queue 106. When scheduler 108 determines that there are sufficient tokens in the token system 110, then a packet may be transferred from the input queue 104 to the output queue 106. Thereafter, packets can be passed in order from the output queue 108 to other portions of the network appliance by forwarding engine 120. At a predetermined rate, increment engine 112 adds tokens to the token system 110. Similarly, as packets are transferred from the input queue 104 to the output queue 106, decrement engine 114 removes the appropriate amount of tokens from the token system 110.

Token system 110 includes a plurality of traffic shaping policy buckets. As described above, a traffic shaping policy is a convention used to ensure levels of service to users of the network appliance. A policy is any user-defined criteria for grouping input received at the network appliance. The policy can be based on IP address or other constraints in order to manage the traffic from one particular user defined group. One or more policies can be evaluated for each packet. For example, one policy may screen based on an IP address for the packet sender. Another policy may screen based on a destination IP address. Other parameters other than IP address can be used in constructing a traffic shaping policy.

In one implementation, for each policy, token system 110 includes at least three buckets 130a, 130b and 130c. Each first bucket 130a, also referred to as a maximum bandwidth allocation bucket, is configured as a conventional token bucket that has a SIR that is equal to the maximum rate at which data of the policy type can be sent. Tokens are added to the first bucket 130a at a rate to limit the bandwidth allocated for the given policy to the maximum bandwidth allocation assigned to the given policy. If the balance (in tokens) in the maximum bandwidth bucket 130a is greater than or equal to zero, then the scheduler 108 will allow the packet to pass from the input queue 104 to the output queue 106 (assuming that the balances in the second and third buckets are appropriate). However, if the balance in the maximum bandwidth bucket 130a is less than zero, then no packets of this type (that include the policy parameter) will pass from the input queue 104 to the output queue 106 at this time. This guarantees that the particular policy cannot be oversubscribed.

Second bucket 130b is a guaranteed bandwidth bucket. The guaranteed bandwidth bucket 130b operates in a similar fashion to the maximum bandwidth bucket 130a and includes an SIR set to be the guaranteed bandwidth allocation for the policy. When a packet is received (assuming that the maximum bandwidth allocation bucket has a balance that is greater than or equal to 0), then the balance of the guaranteed bandwidth bucket 130b is checked to determine if the packet's size is greater than or less than the balance in the guaranteed bandwidth bucket 130b. If the balance of tokens is greater than the packet's size, then scheduler 108 allows the packet to be transferred from the input queue 104 to the output queue 106. If the balance is less than the packet's size, then an attempt is made to borrow from the third bucket 130c.

The third bucket is referred to as the shared bandwidth bucket 130c. The shared bandwidth bucket 130c has tokens added every time one or more of the guaranteed bandwidth buckets reaches its peak allocation. At a next time one or more tokens are scheduled to be added to an otherwise full guaranteed bandwidth bucket 130b (tokens which would otherwise be discarded), the token(s) are stored in the shared bandwidth bucket 130c. In one implementation, the shared bandwidth bucket 130c is shared among plural members (guaranteed bandwidth buckets 130b) of the same traffic shaping policy group. Traffic shaping policies can be grouped together by many different criteria such as physical interface and priority. Policies can also be grouped in a hierarchical way such that there are subgroups in a group. In operation, if the balance of a guaranteed bandwidth bucket 130b for any member of a policy group is less than the packet size, an attempt is made to borrow from the shared bandwidth bucket 130c. If the borrowing attempt fails, that is, if there are insufficient tokens in the shared bandwidth bucket 130c, then the packet will not be transferred to the output queue 106. Alternatively, if there are sufficient tokens in the shared bandwidth bucket 130c to allow for a borrowing operation, then a sufficient number of credits are transferred to the guaranteed bandwidth bucket that attempted the borrowing. Thereafter, the scheduler enables the transfer of the packet from the input queue 104 to the output queue 106. After the transfer, the decrement engine 114 removes the appropriate number of tokens from the shared bandwidth bucket 130c (i.e., the number borrowed) and from the appropriate guaranteed bandwidth bucket 130b (i.e., the number of tokens required to pass the packet).

Referring now to FIG. 2, a process used by the scheduler 108 for determining when to transfer packets from the input queue 104 to the output queue 106 is shown. The process begins as the scheduler checks to determine if any packets are stored in the input queue (202). If not, the process waits for a packet to arrive and continues at step 202. Otherwise, a next packet in the input queue is identified (204). A check is made to determine if a policy screen should be applied to the packet (206). As described above, a policy screen includes one or more policy parameters. The policy parameters define the screening parameters for the policy. An example of a policy parameter is the packet's IP address. Packets can be screened based on destination, source or both. In addition, traffic can be screened based upon protocol type, UPD/TCP port number, type of service and traffic content such as

websites. If no policy screen is to be applied, then the process continues at step 250 where the packet is transferred from the input queue 104 to the output queue 106.

If a policy is to be enforced, a check is made to determine if the size of the packet exceeds the balance of the maximum bandwidth bucket 130a associated with the identified policy (208). If the size exceeds the balance, then the process continues at step 202 without transferring the packet from the input queue 104 to the output queue 106. If the size does not exceed the balance in step 208, the scheduler checks to determine if the size of the packet exceeds the balance of the guaranteed bandwidth bucket 130b for the policy (210). If the size does not exceed the balance in step 210, then the process continues at step 250 where the packet is transferred from the input queue 104 to the output queue 106.

If the size exceeds the balance in step 210, then the scheduler determines if there are sufficient tokens in the shared bandwidth bucket 130c associated with the policy to pass the packet (212). If there are insufficient tokens in the shared bandwidth bucket 130c, the process continues at step 202 without transferring the packet from the input queue 104 to the output queue 106. If there are sufficient tokens in the shared bandwidth bucket 130c, then the appropriate amount of tokens is transferred (so that the balance in the guaranteed bandwidth bucket 130b equals or exceeds the size of the packet) from the shared bandwidth bucket 130c to the requesting guaranteed bandwidth bucket 130b (214). In one implementation, the tokens are physically transferred from the shared bandwidth bucket to the guaranteed bandwidth bucket during a borrowing operation. Alternatively, the tokens can be virtually passed, that is, allocated to the guaranteed bandwidth bucket without requiring them to be physically transferred. A check is made to determine if the packet needs to be screened against any other policies (216), if so, the next policy is identified (218) and the process continues at step 208.

In step 250, the scheduler 108 allows the transfer of the packet from the input queue 104 to the output queue 106. At the time of transfer, the appropriate number of tokens is decremented from the guaranteed bandwidth bucket and maximum bandwidth bucket for each policy used to screen the packet (252). Thereafter the process continues at step 202.

Referring now to FIG. 3, a flow diagram for a method for incrementing the guaranteed bandwidth buckets 130b is shown. The process begins by identifying a guaranteed bandwidth for the policy associated with the given guaranteed bandwidth bucket

(302). A frequency for incrementing (304) and an amount of tokens to be added at each increment step are identified (306). A check is then made to determine if the guaranteed bandwidth bucket should be updated (based on the frequency data) (308). If not, the process continues at step 308 waiting for the next update period. If an update is warranted, then a check is made to determine if there is sufficient space in the guaranteed bandwidth bucket to accept the amount of tokens to be added (310). If there is enough space, then the tokens are added to the guaranteed bandwidth bucket (312) and the process continues at step 308.

If the check at step 310 fails, then the guaranteed bandwidth bucket is filled to its capacity (314). Thereafter, a check is made to determine if the guaranteed bandwidth bucket has an associated shared bandwidth bucket (316). If not, the process continues at step 308. If the guaranteed bandwidth bucket has an associated shared bandwidth bucket then a check is made to determine there is sufficient space in the shared bandwidth bucket to accept the amount of excess tokens (318). If there is not enough space, then the shared bandwidth bucket is filled to its capacity (320) and the process continues at step 308. If there is enough space, then all of the excess tokens are added to the shared bandwidth bucket (322) and the process continues at step 308.

Credit/Debit Guaranteed Bandwidth Bucket

In one implementation, the guaranteed bandwidth bucket 130b can be a credit/debit bucket. In this implementation, the maximum bandwidth bucket 130a is a conventional token bucket (i.e., no-debt bucket). The guaranteed bandwidth bucket 130b is a credit/debit bucket that is allowed to operate in a debt mode. The shared bandwidth bucket 130c is a conventional token (i.e., no-debt) bucket that includes the borrowing provisions set forth above.

In operation, the maximum bandwidth bucket 130a operates (is incremented and decremented) as described above to ensure the policy is not oversubscribed. Assuming that the number of tokens in the maximum bandwidth bucket is greater than or equal to zero, then the scheduler checks the guaranteed bandwidth bucket (the credit/debit version) to determine if the number of tokens is greater than the packet size. If so, then the packet is passed in the system and the appropriate number of tokens is debited from both the maximum bandwidth bucket and the guaranteed bandwidth bucket. If the number of tokens in the guaranteed

bandwidth bucket 130b is greater than 0 however, less than the packet size then the packet passes, but there is no borrowing operation that occurs. The packet passes in accordance with the standard credit/debit protocol associated with a credit/debit bucket. If however the number of tokens in the guaranteed bandwidth bucket 130b is less than 0, the scheduler attempts to borrow (from an associated shared bandwidth bucket 130c, if any).

In one implementation, the scheduler may borrow only to cover the debt (incurred for passing the packet). Alternatively, in another implementation, the scheduler may borrow to cover the packet size itself. In a third implementation, the scheduler may borrow to cover both the debt as well as the packet size.

Two-bucket System

In one implementation, token system 110 includes only cascaded pairs of buckets. In this implementation, token system 110 includes a plurality of guaranteed bandwidth buckets 130b, one or more for each policy that is supported, and one or more shared bandwidth buckets (to allow sharing among policies or in a policy). Each of the guaranteed bandwidth buckets operates as described above. The guaranteed bandwidth buckets can either be conventional token buckets, leaky buckets, or credit/debit buckets. When a guaranteed bandwidth bucket does not have sufficient tokens/space to allow a packet to pass, the scheduler can attempt to borrow from a shared bandwidth bucket associated with a given policy (or policies).

The shared bandwidth bucket can support plural different policies. The network appliance can provide a dynamic allocation of resources that supports both guaranteed bandwidth among policies and also a sharing of guaranteed bandwidth among members of one or more policies.

Priority Information

In an alternative implementation, when a guaranteed bandwidth bucket attempts to borrow from a shared bandwidth bucket, the scheduler considers priority data when determining whether borrowing is allowed. Each guaranteed bandwidth bucket can be assigned a priority. The priority data can be used to determine when borrowing is allowed. The priority data can be set based on use of the shared bandwidth bucket. Depending on the

priority level setting for a given guaranteed bandwidth bucket, the borrowing operation may be disallowed. The priority level can be used to determine how a policy/policies will share the bandwidth in the shared bucket that is available. The higher priority policy can be allocated the shared bandwidth before a lower priority policy.

The present invention can be used by any traffic shaper to enforce traffic shaping policies. The traffic shaper can be a dedicated device such as a package shaper or a multi-function device with traffic shaping functionality such as a router or a firewall engine. The present invention allows a traffic shaper to dynamically allocate guaranteed bandwidth for a policy. System administrators can avoid the limitations of static allocations while guaranteed bandwidth is not compromised

A number of embodiments of the invention have been described. Nevertheless, it will be understood that various modifications may be made without departing from the spirit and scope of the invention. Accordingly, other embodiments are within the scope of the following claims.